АК-ВС 2

Руководство пользователя

Аннотация

«AK-BC 2» анализирует исходный текст программы на языках C/C++/C#, Java и PHP. Руководство пользователя описывает системные требования, состав функций программы и стадии их выполнения.

Содержание

1 Назначение программы

«Анализатор исходных текстов «АК-ВС 2» анализирует исходный текст программ и генерирует следующие отчеты тестовых испытаний:

Отчеты по статическому анализу:

1) Отчёт по метрикам;
2) Список файлов проекта;
3) Список информационных объектов (ИО);
4) Перечень функциональных объектов (ФО) (функций и процедур);
5) Перечень функциональных объектов (ветвей);
6) Перечень невызываемых ФО;
7) Перечень неопределенных ФО;
8) Таблица связей ФО по управлению;
9) Маршруты выполнения ФО;
10) Таблица связей ФО по информации;
11) Критические маршруты для выбранного ИО;
12) Таблица связей функций и ветвей;
13) Маршруты выполнения ФО с ветвями;
14) Блок-схемы ФО;
15) Отчет о поиске потенциально опасных конструкций.

Отчеты по динамическому анализу:

1) Отчёт по метрикам;
2) Отработавшие ФО (процедуры и функции);
3) Отработавшие связи между ФО (процедурами и функциями);
4) Отработавшие ФО (ветви);
5) Отработавшие связи между ФО (процедурами, функциями и ветвями).

2 Системные требования

ТАБЛИЦА 1

Системные требования

Процессор	64-битный Intel-совместимый
Оперативная память	2 ГБ (Рекомендуется 4 Гб)
Жесткий диск (свободно)	30 ГБ (Рекомендуется 100Гб)
Порты	USB-порт для ключа защиты
Поддерживаемые браузеры	Internet Explorer версии 10 Google Chrome версии 30 Mozilla Firefox версии 25
Операционная система	Linux Ubuntu 14.04 Server LTS x64 Microsoft Windows 7 SP1 x64 Microsoft Windows 8 x64 Microsoft Windows Server 2008 x64
Дополнительное ПО	Программная платформа .Net Framework 4.0 (Только для ОС семейства Windows) Java Runtime Environment 7 CodeMeter Runtime 4.5
Возможность запуска виртуальных машин Virtual Box	

3 Работа программы

3.1 Установка

Для виртуальной машины

Дистрибутив представляет собой образ формата OFV.

Импортируйте образ в виртуальную машину.

Для Windows

Дистрибутив представляет собой установочный файл.

Запустите установочный файл и следуйте дальнейшим иструкциям мастера установки.

3.2 Запуск

Для виртуальной машины

Запустите образ программы в виртуальной машине при вставленном USB-ключе CodeMeter.

!!! При отсутствии ключа программа запущена не будет.

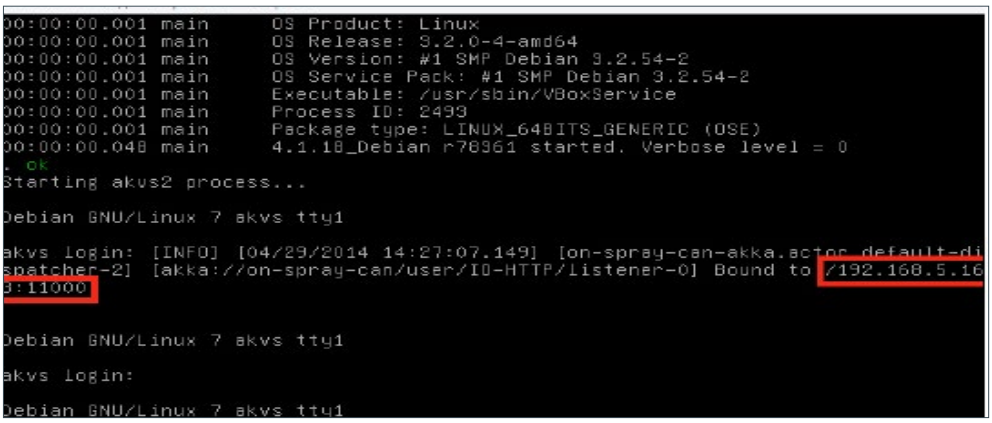
Для Windows

Запустите start.bat.

После запуска «АК-ВС 2», в консоли отобразится IP-адрес, выделенный под веб-интерфейс (рис. 1).

Чтобы получить доступ к веб-интерфейсу «АК-ВС 2», перейдите по адресу, который будет отображен в консоли.

РИСУНОК 1// Пример запуска образа “АК-ВС 2” с выделением IP-адреса



«АК-ВС 2» использует:

- Порт: 11000.
- Логин: admin
- Пароль: admin

Вкладка «**Проекты**» показывает все текущие проекты (рис.2).

РИСУНОК 2 // Вкладка «Проекты» веб-интерфейса

Список проектов

№	Имя	Статус
1	OpenSSL	Статический анализ завершен
2	I2P	Статический анализ завершен
3	sasa	Статический анализ завершен
4	OpenRA	Статический анализ завершен
5	EMpty	Статический анализ завершен
6	sc_test	Статический анализ завершен
7	cs_test	Ошибка статического анализа

- На вкладке «№» — идентификаторы проектов;
- В поле «Имя» — имя проекта;
- В поле «Статус» — стадия, на которой находится проект.

Вкладка «**Датчики**» показывает имеющиеся датчики для динамического анализа (рис.3).

РИСУНОК 3// Вкладка «Датчики» веб-интерфейса

Проекты Датчики О программе Выход

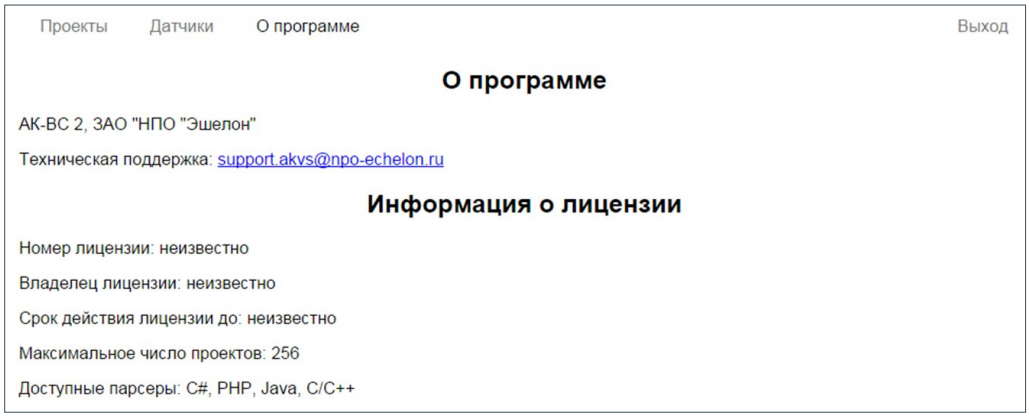
Список типовых датчиков

Файл	Описание
_akvs_probe.cs	Стандартный датчик C#
_akvs_probe.php	Стандартный датчик PHP
_akvs_probe_thread_win32.c	Стандартный датчик для многопоточных WIN32 приложений C/C++
_akvs_probe_pthread.c	Стандартный датчик для многопоточных POSIX приложений C/C++
_akvs_probe.c	Стандартный датчик для однопоточных приложений C/C++
_akvs_probe.h	Стандартный заголовочный файл для датчиков C/C++
JavaProbe.java	Стандартный датчик Java

- В поле «Файл» — имя файла;
- В поле «Описание» — описание файлов-датчиков.

Вкладка «**О программе**» содержит адрес электронной почты технической поддержки продукта и информацию о лицензии (рис.4).

РИСУНОК 4// Вкладка «О программе»



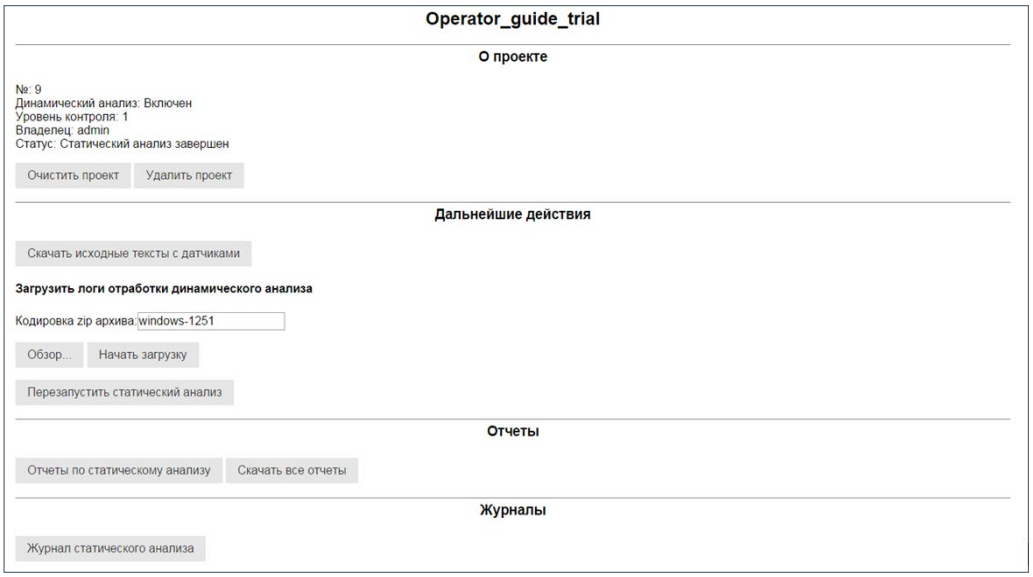
Чтобы выйти из программы, нажимите «**Выход**» в правом верхнем углу.

4 Главные функции

4.1 Просмотр информации о проекте

Выберите нужный проект и нажмите левую кнопку мыши во вкладке «**Проекты**», чтобы постомреть информацию о проекте.

РИСУНОК 5// Страница проекта



- В разделе «О проекте» — номер проекта в списках проектов, уровень контроля, статус проекта;
- В разделе «Дальнейшие действия» — предложение провести динамический анализ кода;
- В разделе «Отчеты» — возможность просмотреть/скачать отчеты;
- В разделе «Журнал» — возможность изучить журнал анализа.

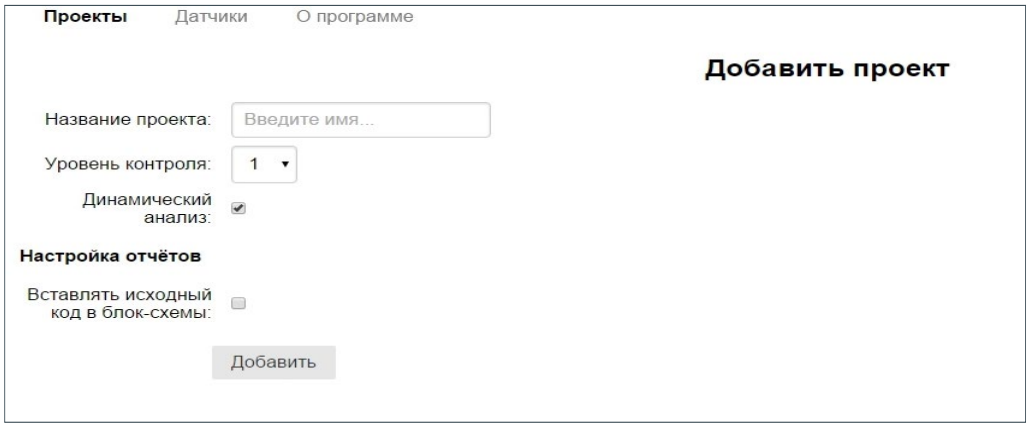
4.2 Добавление нового проекта

Во вкладке «**Проекты**» нажмите кнопку «**Добавить**».

Рисунок 6 показывает, как настоить новый проект.

.

РИСУНОК 6//Добавление нового проекта



- В поле «**Имя**» введите имя проекта;
- В поле «**Уровень контроля**» выберите уровень контроля в соответствии с РД «Защита от несанкционированного доступа к информации. Часть 1. Программное обеспечение средств защиты информации. Классификация по уровню контроля отсутствия недекларированных возможностей»;
- Включите опцию «**Динамический анализ**», если требуется выполнить его после проведения статического анализа;
- Включите опцию «**Вставлять исходный код в бок-схемы**», если требуется, чтобы исходный код был вставлен в блок-схемы при составлении отчетов.
- Нажмите «**Добавить**».

4.3 Очистка проекта

В разделе **«О проекте»** нажмите кнопку «**Очистить проект**», чтобы удалить все файлы с исходным текстом и все отчеты.

!!! ВНИМАНИЕ! Это действие нельзя отменить.\

4.4 Удаление проекта

В разделе **«О проекте»** нажмите кнопку «**Удалить проект**», чтобы удалить сам проект.

!!! ВНИМАНИЕ! Это действие нельзя отменить.

5 Статический анализ

5.1 Начало статического анализа

1. **Загрузите в проект исходный текст, который будет анализироваться.**

Для этого необходимо выбрать проект во вкладке «Проекты» и в разделе «Дальнейшие действия» нажать кнопку «Обзор…».

2. **Выберите исходный текст для загрузки.**

Исходные тексты должны находиться в архиве с расширением .zip. При необходимости имеется возможность задать кодировку zip-архива.

3. **Нажмите кнопку «Начать загрузку».**

Статус проекта при этом должен измениться на «Исходные тексты загружены».

4. **Выберите конфигурацию парсера,** то есть программной части, выполняющей синтаксический анализ.

Конфигурация парсера должна находиться в архиве с расширением .zip

Также можно пропустить этап загрузки конфигурации парсера. Для этого нажмите на кнопку **«Пропустить этап конфигурации»**.

При этом статус проекта должен измениться на «Выполнение статического анализа».

5.2 Окончание статического анализа

Дождитесь окончания статического анализа. При необходимости нажмите кнопку «Обновить».

При окончании анализа статус проекта должен измениться на «Статический анализ завершен».

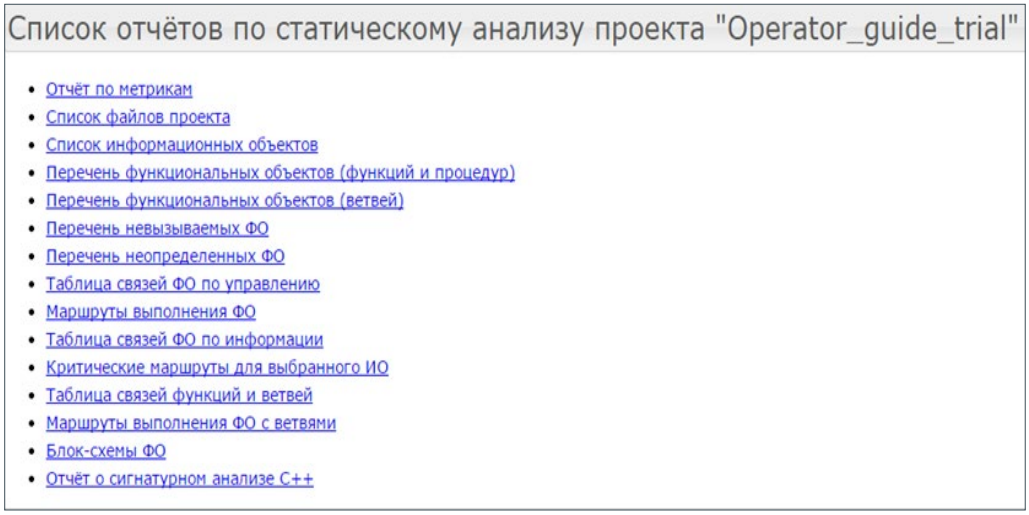
После этого на странице проекта появятся разделы **«Отчеты»** и **«Журналы»**.

В разделе **«Отчеты»** нажмите кнопку **«Отчеты по статическому анализу»**. После этого откроется страница с отчетами по статическому анализу как на рисунке 7.

5.3 Примеры отчетов

На рисунке 7 представлена **страница с отчетами по статическому анализу.** Для просмотра интересующего отчета и нажмите левую кнопку мыши. Примеры отчетов представлены на рисунках 8 – 26.

РИСУНОК 7//Страница с отчетами по статическому анализу



Список отчётов по статическому анализу проекта "Operator_guide_trial"

- Отчёт по метрикам
- Список файлов проекта
- Список информационных объектов
- Перечень функциональных объектов (функций и процедур)
- Перечень функциональных объектов (ветвей)
- Перечень невызываемых ФО
- Перечень неопределенных ФО
- Таблица связей ФО по управлению
- Маршруты выполнения ФО
- Таблица связей ФО по информации
- Критические маршруты для выбранного ИО
- Таблица связей функций и ветвей
- Маршруты выполнения ФО с ветвями
- Блок-схемы ФО
- Отчёт о сигнатурном анализе C++

Рисунок 8 содержит отчет по **метрикам статического анализа.**

РИСУНОК 8//Отчет по метрикам статического анализа

Operator_guide_trial: отчёт по метрикам статического анализа

Список кодировок в проекте	ASCII
Список языков в проекте	C/C++
Общее количество файлов	14
Суммарный размер файлов	527 KB (539 139 байт)
Средний размер файла	38 KB (38 509 байт)
Количество строк кода	14 849
Среднее количество строк в файле	1 060
Количество информационных объектов	1 700
Количество функциональных объектов	279
Количество ветвей	399
Количество невызываемых функциональных объекто	243
Количество неопределенных функциональных объек	150
Количество связей по управлению	51
Количество связей по информации	87
Количество связей функций и ветвей	450

На рисунке 9 представлен **список файлов проекта.** В отчете имеются следующие поля: **№, Путь до файла, Язык программирования и Кодировка файла.**

РИСУНОК 9// Список файлов проекта

Operator_guide_trial: список файлов проекта

№	Путь до файла	Язык программирования	Кодировка файла
0	home/denis/memcached/timedrun.c	CPP	ASCII
1	home/denis/memcached/testapp.c	CPP	ASCII
2	home/denis/memcached/util.c	CPP	ASCII
3	home/denis/memcached/items.c	CPP	ASCII
4	home/denis/memcached/_akvs_probe.h	CPP	ASCII
5	home/denis/memcached/thread.c	CPP	ASCII
6	home/denis/memcached/hash.c	CPP	ASCII
7	home/denis/memcached/slabs.c	CPP	ASCII
8	home/denis/memcached/daemon.c	CPP	ASCII
9	home/denis/memcached/assoc.c	CPP	ASCII
10	home/denis/memcached/cache.c	CPP	ASCII
11	home/denis/memcached/memcached.c	CPP	ASCII
12	home/denis/memcached/stats.c	CPP	ASCII
13	home/denis/memcached/sizes.c	CPP	ASCII

На рисунке 10 представлен **список информационных объектов.** В отчете имеются следующие поля: **№, QID (Qualified ID), Название и Расположение.**

РИСУНОК 10// Список информационных объектов

Operator_guide_trial: список информационных объектов

№	QID	Название	Расположение
1	0:0	caught	home/denis/memcached/timedrun.c:11
2	0:2	caught_signal::which	home/denis/memcached/timedrun.c:13
3	0:3	caught_signal::_akvs_probe_1_207	home/denis/memcached/timedrun.c:14
4	0:5	wait_for_process::pid	home/denis/memcached/timedrun.c:20
5	0:6	wait_for_process::_akvs_probe_3_271	home/denis/memcached/timedrun.c:21
6	0:7	wait_for_process::rv	home/denis/memcached/timedrun.c:23
7	0:8	wait_for_process::stats	home/denis/memcached/timedrun.c:24
8	0:9	wait_for_process::i	home/denis/memcached/timedrun.c:25
9	0:10	wait_for_process::sig_handler	home/denis/memcached/timedrun.c:26
10	0:12	wait_for_process::_akvs_probe_9_739	home/denis/memcached/timedrun.c:38
11	0:13	wait_for_process::p	home/denis/memcached/timedrun.c:40
12	0:15	wait_for_process::_akvs_probe_11_806	home/denis/memcached/timedrun.c:41
13	0:16	wait_for_process::FI::__in	home/denis/memcached/timedrun.c:44
14	0:17	wait_for_process::FI::__i	home/denis/memcached/timedrun.c:44
15	0:18	wait_for_process::FI::__in	home/denis/memcached/timedrun.c:45
16	0:19	wait_for_process::FI::__i	home/denis/memcached/timedrun.c:45
17	0:20	wait_for_process::FI::__in	home/denis/memcached/timedrun.c:46
18	0:21	wait_for_process::FI::__i	home/denis/memcached/timedrun.c:46
19	0:23	wait_for_process::_akvs_probe_18_1012	home/denis/memcached/timedrun.c:50
20	0:24	wait_for_process::sig	home/denis/memcached/timedrun.c:52
21	0:26	wait_for_process::_akvs_probe_20_1169	home/denis/memcached/timedrun.c:56
22	0:28	wait_for_process::_akvs_probe_21_1246	home/denis/memcached/timedrun.c:58
23	0:30	wait_for_process::_akvs_probe_22_1395	home/denis/memcached/timedrun.c:67
24	0:33	wait_for_process::_akvs_probe_24_1559	home/denis/memcached/timedrun.c:77

На рисунке 11 представлен **перечень функциональных объектов (функций и процедур)**. В отчете имеются поля, аналогичные представленным в списке информационных объектов.

РИСУНОК 11// Перечень функциональных объектов(функций и процедур)

Operator_guide_trial: перечень функциональных объектов (функций и процедур)

№	QID	Название	Расположение
1	0:1	caught_signal	home/denis/memcached/timedrun.c:13
2	0:4	wait_for_process	home/denis/memcached/timedrun.c:20
3	0:34	spawn_and_wait	home/denis/memcached/timedrun.c:92
4	0:44	main	home/denis/memcached/timedrun.c:120
5	1:4	cache_create_test	home/denis/memcached/testapp.c:36
6	1:7	cache_constructor	home/denis/memcached/testapp.c:48
7	1:13	cache_constructor_test	home/denis/memcached/testapp.c:55
8	1:17	cache_fail_constructor	home/denis/memcached/testapp.c:68
9	1:22	cache_fail_constructor_test	home/denis/memcached/testapp.c:73
10	1:29	cache_destructor	home/denis/memcached/testapp.c:93
11	1:33	cache_destructor_test	home/denis/memcached/testapp.c:99
12	1:36	cache_reuse_test	home/denis/memcached/testapp.c:112
13	1:43	cache_bulkalloc	home/denis/memcached/testapp.c:132
14	1:53	test_issue_161	home/denis/memcached/testapp.c:159
15	1:58	cache_redzone_test	home/denis/memcached/testapp.c:172
16	1:64	test_safe_strtoul	home/denis/memcached/testapp.c:208
17	1:67	test_safe_strtoull	home/denis/memcached/testapp.c:233
18	1:70	test_safe_strtoll	home/denis/memcached/testapp.c:253
19	1:73	test_safe_strtol	home/denis/memcached/testapp.c:283
20	1:76	start_server	home/denis/memcached/testapp.c:322
21	1:103	test_issue_44	home/denis/memcached/testapp.c:465
22	1:107	lookuphost	home/denis/memcached/testapp.c:476
23	1:121	connect_server	home/denis/memcached/testapp.c:504
24	1:138	test_vperror	home/denis/memcached/testapp.c:549
25	1:149	send_ascii_command	home/denis/memcached/testapp.c:594

На рисунке 12 представлен **перечень функциональных объектов (ветвей)**. В отчете имеются поля, аналогичные представленным в списке информационных объектов.

РИСУНОК 12// Перечень функциональных объектов(ветвей)

Operator_guide_trial: перечень функциональных объектов (ветвей)

№	QID	Название	Расположение
1	0:11	for	home/denis/memcached/timedrun.c:38
2	0:14	if	home/denis/memcached/timedrun.c:41
3	0:22	if	home/denis/memcached/timedrun.c:50
4	0:25	switch	home/denis/memcached/timedrun.c:56
5	0:27	if	home/denis/memcached/timedrun.c:58
6	0:29	switch	home/denis/memcached/timedrun.c:67
7	0:31	switch	home/denis/memcached/timedrun.c:73
8	0:32	if	home/denis/memcached/timedrun.c:77
9	0:39	switch	home/denis/memcached/timedrun.c:100
10	0:41	switch	home/denis/memcached/timedrun.c:107
11	0:43	switch	home/denis/memcached/timedrun.c:115
12	1:26	if	home/denis/memcached/testapp.c:82
13	1:40	for	home/denis/memcached/testapp.c:120
14	1:48	for	home/denis/memcached/testapp.c:140
15	1:51	for	home/denis/memcached/testapp.c:148
16	1:56	if	home/denis/memcached/testapp.c:163
17	1:85	if	home/denis/memcached/testapp.c:347
18	1:90	if	home/denis/memcached/testapp.c:374
19	1:92	while	home/denis/memcached/testapp.c:396
20	1:95	if	home/denis/memcached/testapp.c:403
21	1:98	while	home/denis/memcached/testapp.c:413
22	1:100	if	home/denis/memcached/testapp.c:415
23	1:115	if	home/denis/memcached/testapp.c:487
24	1:117	if	home/denis/memcached/testapp.c:489
25	1:119	if	home/denis/memcached/testapp.c:493
26	1:128	if	home/denis/memcached/testapp.c:509

На рисунке 13 представлен **перечень невызываемых ФО (функциональных объектов)**. В отчете имеются поля, аналогичные представленным в списке информационных объектов.

РИСУНОК 13// Перечень невезываемых ФО (Функциональных объектов)

Operator_guide_trial: перечень невызываемых ФО

№	QID	Название	Расположение
1	0:1	caught_signal	home/denis/memcached/timedrun.c:13
2	0:44	main	home/denis/memcached/timedrun.c:120
3	1:4	cache_create_test	home/denis/memcached/testapp.c:36
4	1:7	cache_constructor	home/denis/memcached/testapp.c:48
5	1:13	cache_constructor_test	home/denis/memcached/testapp.c:55
6	1:17	cache_fail_constructor	home/denis/memcached/testapp.c:68
7	1:22	cache_fail_constructor_test	home/denis/memcached/testapp.c:73
8	1:29	cache_destructor	home/denis/memcached/testapp.c:93
9	1:33	cache_destructor_test	home/denis/memcached/testapp.c:99
10	1:36	cache_reuse_test	home/denis/memcached/testapp.c:112
11	1:43	cache_bulkalloc	home/denis/memcached/testapp.c:132
12	1:53	test_issue_161	home/denis/memcached/testapp.c:159
13	1:58	cache_redzone_test	home/denis/memcached/testapp.c:172
14	1:64	test_safe_strtoul	home/denis/memcached/testapp.c:208
15	1:67	test_safe_strtoull	home/denis/memcached/testapp.c:233
16	1:70	test_safe_strtoll	home/denis/memcached/testapp.c:253
17	1:73	test_safe_strtol	home/denis/memcached/testapp.c:283
18	1:76	start_server	home/denis/memcached/testapp.c:322
19	1:103	test_issue_44	home/denis/memcached/testapp.c:465

На рисунке 14 представлен **перечень неопределенных ФО**. В отчете имеются следующие поля: **№, Название** и **Количество вызовов.** При этом, если нажать символ «+» в крайней левой колонке, то появится выпадающий список результатов, сгруппированный по неопределенным объектам. В выпадающем списке имеются следующие поля: **№, QID вызывающего ФО, Название вызывающего ФО и Файл и строка с вызовом.**

РИСУНОК 14// Перечень неопределенных ФО

Operator_guide_trial: перечень неопределенных ФО

	№	Название	Количество вызовов
+	1	APPEND_NUM_STAT()	8
−	2	APPEND_STAT()	7

№	QID вызывающего ФО	Название вызывающего ФО	Файл и строка с вызовом
1	0:101	do_slabs_stats	home/denis/memcached/slabs.c:461
2	0:101	do_slabs_stats	home/denis/memcached/slabs.c:462
3	6:89	do_item_stats_totals	home/denis/memcached/items.c:539
4	6:89	do_item_stats_totals	home/denis/memcached/items.c:541
5	6:89	do_item_stats_totals	home/denis/memcached/items.c:543
6	6:89	do_item_stats_totals	home/denis/memcached/items.c:545
7	6:104	do_item_stats_sizes	home/denis/memcached/items.c:636

Стр. 1 из 1

+	3	MEMCACHED_ASSOC_FIND()	1
+	4	MEMCACHED_SLABS_ALLOCATE()	1
+	5	MEMCACHED_SLABS_ALLOCATE_FAILED()	1
+	6	MEMCACHED_SLABS_FREE()	1
+	7	MEMCACHED_SLABS_SLABCLASS_ALLOCATE()	1
+	8	MEMCACHED_SLABS_SLABCLASS_ALLOCATE_FAILED()	1
+	9	STATS_LOCK()	12
+	10	STATS_UNLOCK()	13
+	11	__assert_fail(const char *, const char *, unsigned int, const char *)	96
+	12	__builtin_va_end(__va_list_tag *)	2
+	13	__builtin_va_start(__va_list_tag *, ...)	2
+	14	__ctype_b_loc()	5
+	15	__errno_location()	22
+	16	_akvs_probe()	1463

На рисунке 15 представлена **таблица связей ФО по управлению**. В отчете имеются следующие поля: **№, QID вызывающего ФО, Название вызывающего ФО, QID вызываемого ФО, Название вызываемого ФО.**

РИСУНОК 15//Таблица связей ФО по управлению

Operator_guide_trial: таблица связей ФО по управлению

№	QID вызывающего ФО	Название вызывающего ФО	QID вызываемого ФО	Название вызываемого ФО
1	0:34	spawn_and_wait	0:4	wait_for_process
2	1:159	test_issue_92	1:149	send_ascii_command
3	1:162	test_issue_102	1:149	send_ascii_command
4	1:173	shutdown_memcached_server	1:149	send_ascii_command
5	1:507	test_issue_101	1:252	test_binary_noop
6	5:104	worker_libevent	5:46	register_thread_initialized
7	5:107	thread_libevent_process	5:46	register_thread_initialized
8	5:107	thread_libevent_process	5:86	cqi_free
9	5:225	thread_init	5:41	wait_for_thread_registration
10	7:28	slabs_preallocate	7:56	do_slabs_newslab
11	7:56	do_slabs_newslab	7:48	split_slab_page_into_freelist
12	7:62	do_slabs_alloc	7:56	do_slabs_newslab
13	7:213	do_slabs_reassign	7:202	slabs_reassign_pick_any
14	11:478	server_socket_unix	11:470	new_socket_unix
15	11:556	main	11:26	stats_init
16	11:556	main	11:30	settings_init
17	11:556	main	11:32	conn_init
18	11:556	main	11:498	clock_handler
19	11:556	main	11:506	usage
20	11:556	main	11:508	usage_license
21	11:556	main	11:510	save_pid

На рисунке 16 представлены **маршруты выполнения ФО**. В отчете представлены поля: **№, QID ФО**, **Название ФО и Маршрут.** Нажите на ссылку, чтобы посмотреть маршрут в интерактивном режиме. После этого откроется страница, изображенная на рисунке 17. Нажмите соответствующую кнопку, чтобы продолжить маршрут вперед или назад. Полученные маршруты можно сохранить в формате SVG.

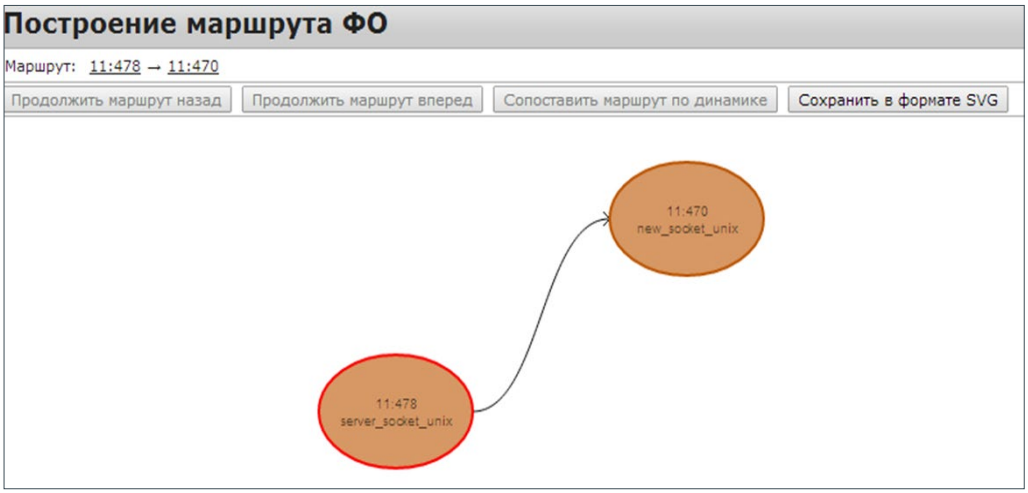
РИСУНОК 16//Маршруты выполнения ФО

Operator_guide_trial: выберите исходный ФО для построения маршрута

№	QID ФО	Название ФО	Маршрут
1	0:33	slabs_preallocate	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
2	0:54	split_slab_page_into_freelist	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
3	0:62	do_slabs_newslab	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
4	0:70	do_slabs_alloc	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
5	0:133	slabs_alloc	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
6	0:138	slabs_free	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
7	0:241	do_slabs_reassign	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
8	4:53	test_issue_161	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
9	4:169	test_issue_92	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
10	4:172	test_issue_102	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
11	4:186	shutdown_memcached_server	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
12	4:352	test_binary_replace_impl	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
13	4:372	test_binary_delete_impl	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
14	4:413	test_binary_getq_impl	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
15	4:498	test_binary_concat_impl	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
16	4:549	test_binary_pipeline_hickup	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
17	4:561	test_issue_101	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
18	7:34	spawn_and_wait	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
19	10:183	process_bin_stat	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
20	10:368	process_stat	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
21	10:408	process_get_command	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
22	10:578	server_socket_unix	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
23	10:656	main	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
24	11:37	stats_prefix_record_get	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО

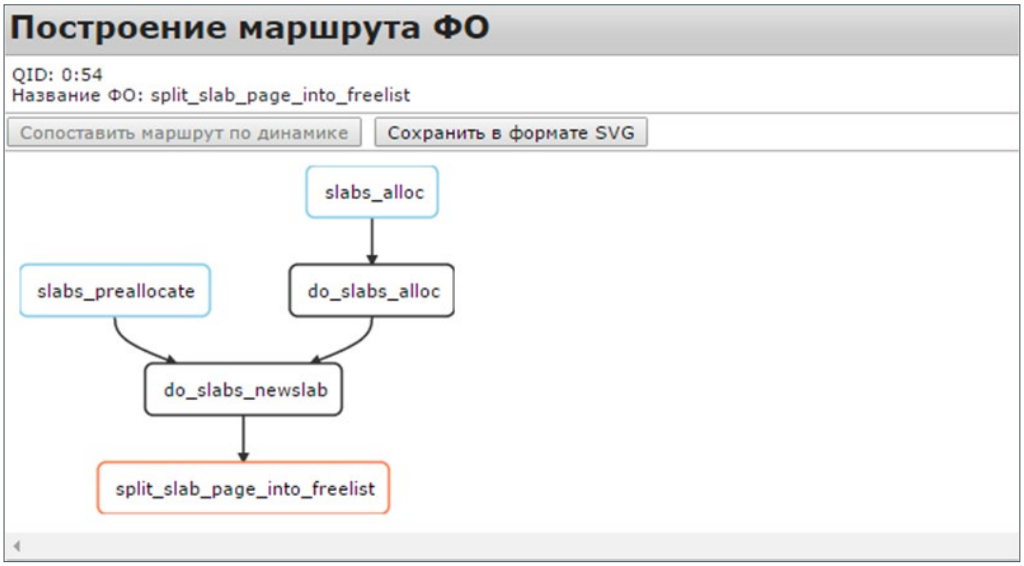
На рисунке 17 препдоставлен пример **построения маршртутов от точек выхода до выбранного ФО.** Нажмите кнопки **“продолжить маршрут вперед”** или **“продолжить маршрут назад”**, чтобы попасть в соотвествующие меню. Полученные маршруты можно сохранить в формате SVG.

РИСУНОК 17//Пример построения маррутов от точек входа до выбранного ФО



На рисунке 18 показаны все маршруты от точки входа до выбранного ФО. Нажмите на соответствующую ссылку, чтобы просмотреть интересующий маршрут. Маршруты можно сохранить в формате SVG. Также имеется возможность сопоставить маршруты по динамике, если был проведен динамический анализ.

РИСУНОК 18//Пример построения маршрутов от точек входа до выбранного ФО



На рисунке 19 представлена **таблица связей ФО по информации**. В отчете представлены следующие поля: **№, QID ИО, Название ИО, QID ФО и Название ФО**, где ИО – информационный объект, а ФО – функциональный объект.

РИСУНОК 19//Таблица связей по ФО по информации

Operator_guide_trial: таблица связей ФО по информации

№	QID ИО	Название ИО	QID ФО	Название ФО
1	0:0	caught	0:1	caught_signal
2			0:4	wait_for_process
4	1:1	TEST_SKIP	1:507	test_issue_101
5			1:540	main
7			1:4	cache_create_test
8			1:13	cache_constructor_test
9			1:33	cache_destructor_test
10			1:36	cache_reuse_test
11			1:43	cache_bulkalloc
12			1:53	test_issue_161
13			1:58	cache_redzone_test
14			1:64	test_safe_strtoul
15			1:67	test_safe_strtoull
16			1:70	test_safe_strtoll
17			1:73	test_safe_strtol
18			1:103	test_issue_44
19			1:159	test_issue_92
20			1:162	test_issue_102
21			1:167	start_memcached_server
22			1:169	stop_memcached_server
23			1:173	shutdown_memcached_server
24	1:2	TEST_PASS	1:252	test_binary_noop
25			1:258	test_binary_quit_impl
26			1:271	test_binary_set_impl
27			1:290	test_binary_add_impl
28			1:313	test_binary_replace_impl

На рисунке 20 представлен **отчет по критическим маршрутам для выбранного ИО**. В отчете представлены следующие поля: **№, QID ИО и Название ИО**. Выберите один из ИО для просмотра маршрутов, представленных в таблице.

РИСУНОК 20// Отчет по критическим маршрутам

Operator_guide_trial: выберите ИО

№	QID ИО	Название ИО
1	0:0	caught
2	0:2	caught_signal::which
3	0:5	wait_for_process::pid
4	0:7	wait_for_process::rv
5	0:8	wait_for_process::stats
6	0:9	wait_for_process::i
7	0:10	wait_for_process::sig_handler
8	0:13	wait_for_process::p
9	0:24	wait_for_process::sig
10	0:35	spawn_and_wait::argv
11	0:37	spawn_and_wait::rv
12	0:38	spawn_and_wait::pid
13	0:45	main::argc
14	0:46	main::argv

На рисунке 21 представлены **критические маршруты для выбранного ИО.** В отчете представлены следующие поля: **№, QID, Название и Маршрут**. Нажмите на надпись **«Проложить маршрут»** для просмотра маршрута.

РИСУНОК 21//Пример отчета по критическим маршрутам для выбранного ИО

Operator_guide_trial: критические маршруты для выбранного ИО

№	QID	Название	Маршрут
1	1:13	cache_constructor_test	проложить маршрут
2	1:22	cache_fail_constructor_test	проложить маршрут
3	1:33	cache_destructor_test	проложить маршрут
4	1:497	test_binary_pipeline_hickup	проложить маршрут

Стр. 1 из 1 1000 Просмотр 1 - 4 из 4

На рисунке 22 представлена **таблица связей функций и ветвей**. В отчете представлены следующие поля: **№, QID вызывающего объекта, Название вызывающего объекта, QID вызываемого объекта и Название вызываемого объекта.**

РИСУНОК 22//Пример таблицы связей функций и ветвей

Operator_guide_trial: таблица связей функций и ветвей

№	QID вызывающего объекта	Название вызывающего объекта	QID вызываемого объекта	Название вызываемого объекта
1	0:4	wait_for_process	0:11	for
2	0:11	for	0:14	if
3	0:11	for	0:22	if
4	0:22	if	0:25	switch
5	0:22	if	0:29	switch
6	0:22	if	0:31	switch
7	0:22	if	0:32	if
8	0:25	switch	0:27	if
9	0:34	spawn_and_wait	0:39	switch
10	0:34	spawn_and_wait	0:41	switch

На рисунке 23 представлены **маршруты выполнения ФО с ветвями**. В отчете имеются поля, аналогичные представленным в маршрутах выполнения ФО. Чтобы посмотреть интересующий маршрут, нажмите на соответствующую ссылку. Страница для построения интерактивных маршрутов и страница, содержащая все маршруты от точек входа до выбранного ФО, аналогичны аналогичны рисунку 17 и 18.

РИСУНОК 23//Маршруты выполнения ФО с ветвями

Operator_guide_trial: выберите исходный ФО для построения маршрута

№	QID ФО	Название ФО	Маршрут
1	0:17	slabs_clsid	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
2	0:23	slabs_init	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
3	0:33	slabs_preallocate	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
4	0:44	grow_slab_list	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
5	0:54	split_slab_page_into_freelist	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
6	0:62	do_slabs_newslab	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
7	0:70	do_slabs_alloc	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
8	0:101	do_slabs_stats	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО
9	0:119	memory_allocate	Построить маршрут в интерактивном режиме Построить все маршруты от точек входа до выбранного ФО

На рисунке 24 представлена **таблица для построения блок-схемы ФО.** В отчете имеются поля, аналогичные представленным в маршрутах выполнения ФО. Нажмите на один из представленных ФО, для просмотра блок-схемы. После этого откроется страница, показанная на рисунке 25. Представленную блок-схему можно сохранить в формате SVG..

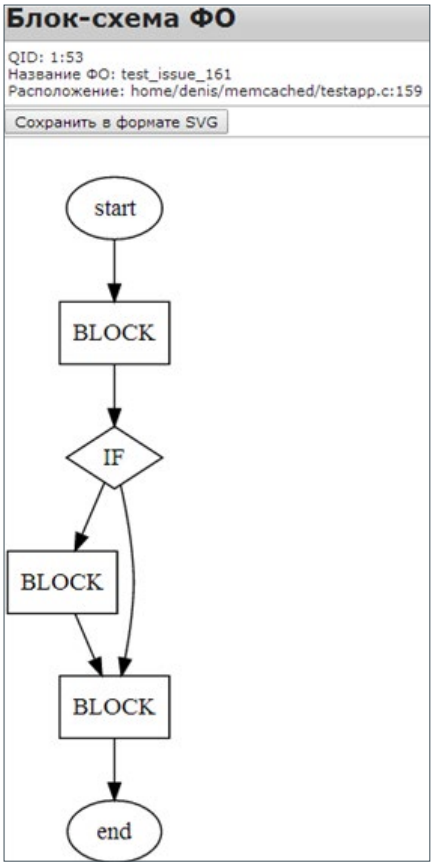
РИСУНОК 24//Таблица выбора ФО для построения блок-схемы

Operator_guide_trial: выберите ФО для построения блок-схемы

№	QID ФО	Название ФО
1	0:1	caught_signal
2	0:4	wait_for_process
3	0:34	spawn_and_wait
4	0:44	main
5	1:4	cache_create_test
6	1:7	cache_constructor
7	1:13	cache_constructor_test
8	1:17	cache_fail_constructor
9	1:22	cache_fail_constructor_test
10	1:29	cache_destructor
11	1:33	cache_destructor_test
12	1:36	cache_reuse_test

РИСУНОК 25

Блок-схема для выбранного ФО



На рисунке 26 представлен **отчет о сигнатурном анализе,** отображаещем дефекты программного кода с сопоставлением им соответствующих CWE-идентификаторов. . В отчете имеются следующие поля: **№, Описание по CWE, Файл, № строки и Строка.**

РИСУНОК 26//Отчет о сигнатурном анализе

Отчёт о сигнатурном анализе

№	Описание по CWE	Файл	№ Строки	Строка
1	Переполнение буфера (CWE:120)	../../data/projects/2/src/popcorn_195_source/source/sc	1413	if (GetAPOPTimeStamp (m_serverReply _ timeStamp)
2	Переполнение буфера (CWE:120)	../../data/projects/2/src/popcorn_195_source/source/sc	1415	sprintf (authString _ %s %s _ GETP_String (PROFILE_
3	Переполнение буфера на стеке (Срыв стека) (CWE:	../../data/projects/2/src/popcorn_195_source/source/sc	2329	strcpy (s _ username) ;
4	Шпионское ПО (CWE:512)	../../data/projects/2/src/popcorn_195_source/source/m	68	if (0 != GetTempPath (MAX_PATH _ m_appTempDir))
5	Вызов сторонних модулей. (CWE: 510)	../../data/projects/2/src/popcorn_195_source/source/m	152	} { m_richEdDLL = LoadLibrary (riched20.dll) ;
6	Переполнение буфера на стеке (Срыв стека) (CWE:	../../data/projects/2/src/popcorn_195_source/source/m	172	strcpy (m_appCmdLine _ commandLine) ;
7	Предопределенный IP-адрес. (CWE: 489)	../../data/projects/2/src/popcorn_195_source/source/se	55	static char * DefaultSpellCheckURL = http://dictionary.
8	Переполнение буфера (CWE:120)	../../data/projects/2/src/popcorn_195_source/source/se	1067	sprintf (buff _ \%s_%d_%08x_%08x _ font-&rt;name
9	Переполнение буфера (CWE:120)	../../data/projects/2/src/popcorn_195_source/source/se	1187	sprintf (section _ %s%d _ m_settingsecs [2] . name
10	Переполнение буфера (CWE:120)	../../data/projects/2/src/popcorn_195_source/source/se	1325	sprintf (section _ %s%d _ m_settingsecs [2] . name
11	Переполнение буфера (CWE:120)	../../data/projects/2/src/popcorn_195_source/source/m	832	if (strlen (m_to)) { if (to) sprintf (m_header + strle
12	Переполнение буфера (CWE:120)	../../data/projects/2/src/popcorn_195_source/source/m	834	else sprintf (m_header + strlen (m_header) _ HF_TO
13	Переполнение буфера (CWE:120)	../../data/projects/2/src/popcorn_195_source/source/m	844	} if (strlen (m_cc)) { if (cc) sprintf (m_header + str

CWE (Common Weakness Enumeration) представляет собой список дефектов безопасности программного обеспечения. CWE служит в качестве базового стандарта для идентификации дефектов, смягчения их последствий и профилактических мероприятий. CWE охватывает более 15000 дефектов безопасности и включает в себя детальную структуру классификации из различных научных источников и примеров.

CWE позволяет компаниям ускорить проверку разрабатываемого программного обеспечения, а так же помогает пользователям в выборе продуктов и услуг, соответствующих их требованиям.

CWE-совместимость позволяет продукту быть зарегистрированным в качестве официально “CWE-совместимого”. CWE-совместимый продукт должен предоставлять пользователям возможность поиска дефектов безопасности по CWE-идентификаторам и получения связанных с ними CWE-идентификаторов.

В сводном отчете:

- Чтобы получить более подробную иформацию о дефетке, нажмите на CWE-идентификатор;
- Чтобы воспользоваться поиском дефектов по CWE-идентификаторам, кликните на значок лупы в левом нижнем углу отчета, выберите параметр поиска «Описание по CWE», укажите условие «содержит» и введите CWE-идентификатор.

Чтобы скачать все отчеты по статическому анализу:

В разделе **«Отчеты»** нажмите кнопку **«Скачать все отчеты».**Отчеты скачиваются в архиве с расширением .zip.

Чтобы посмотреть отчеты статического анализа:

Откройте в браузере файл **index.html.**

Чтобы посмотреть журнал статического анализа:

В разделе **«Журналы»** нажмите кнопку **«Журнал статического анализа».** Журнал выглядит так, как показано на рисунке 27.

РИСУНОК 27//Журнал о статическом анализе

```
Starting project execution at 13:39:33 16/02/2015
JVM memory settings:
Total memory: 62390272
Free memory: 49298912
Maximum memory: 922746880
Launching source loader with arguments -r -e c,cc,cpp,cxx,h,hpp,java,php,cs,plx,plas,pls3 -i ../../data/projects/2/src -p ../../data/projects/2/prj -n popcorn_test at 13:39:33
16/02/2015
Starting import...
Import successfully completed
Launching encoder with arguments -d ../../data/projects/2/prj -m convert at 13:39:33 16/02/2015
Converting files to UTF-8...
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/ecdh.h{0}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/hmac.h{1}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/ripemd.h{2}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/md5.h{3}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/applink.c{4}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/ui_compat.h{5}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/rc2.h{6}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/aes.h{7}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/ssl3.h{8}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/symhacks.h{9}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/ebcdic.h{10}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/rc4.h{11}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/asn1t.h{12}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/dso.h{13}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/ui.h{14}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/cast.h{15}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/buffer.h{17}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/ecdsa.h{16}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/safestack.h{19}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/stack.h{18}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/e_os2.h{21}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/des.h{20}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/bio.h{23}
Converting to UTF-8: ../../data/projects/2/src/popcorn_195_source/OpenSSL/include/openssl/dtls1.h{22}
```

6 Динамический анализ

Динамический анализ проводится после статического анализа.

6.1 Начало динамического анализа

В ходе статического анализа в исходный текст внедряются датчики.

1) Скачайте исходные тексты с датчиками.

В разделе «**Дальнейшие действия**» нажмите кнопку **«Скачать исходные тексты с датчиками»**.

2) Соберите приложение с внедренными датчиками, используя включенные в поставку «АК-ВС 2» исходные файлы датчиков для соответствующего языка.

Необходимые датчики можно скачать во вкладке **«Датчики».**

6.1.1 Внедрение датчиков

6.1.1.1 Сборка проекта C/C++

Расположение датчика: файл _akvs_probe.h

Датчик для C/C++ состоит из двух частей:

- декларация датчика;
- функция датчика (определение функции).

Декларация датчика должна быть в каждом исходном файле, а определение - только в одном из набора файлов, объектные коды которых передаются линковщику.

АК-ВС 2 вставляет в начало каждого файла конструкцию:

```
#include “_akvs_probe.h”
```

а в те файлы, где есть функция main/WinMain, конструкцию:

```
#define _AKVS_PROBE_IMPLEMENTATION_

#include “_akvs_probe.h”
```

!!! Датчик поддерживает многопоточные приложения. Отключите режим многопоточности, если он не используется в программе, удалив из файла _akvs_probe.h строчку #define _AKVS_PROBE_MULTITHREAD

Чтобы собрать проект C/C++:

1. Скачайте датчик _akvs_probe.h.
2. Скопируйте его в include path проекта.

(Для unix-систем можно разместить его в /usr/include, чтобы не копировать датчик в каждый подпроект.)

6.1.1.2 Сборка проекта C#:

 Расположение датчика: файл _akvs_probe.h

Чтобы собрать проект C#

1. Скачайте датчик _akvs_probe.cs.
2. Добавьте его в проект.

6.1.1.3 Сборка проекта Java:

 Расположение датчика: пакет ru.cnpo.akvs2.probejava;

Чтобы собрать проект Java:

1. Создайте путь ru/cnpo/akvs2/probejava в директории с исходными файлами.
2. Скопируйте туда датчик JavaProbe.java

6.2 Обработка результатов

Запустите собранное приложение и проведите максимально глубокое функциональное тестирование. Можно проводить тестирование в несколько этапов, сохраняя получившиеся файлы лога.

Ответственность за полноту тестирования несет человек, проводящий тестирование. По завершении тестирования в рабочей директории приложения будет находиться файл лога отработки датчиков (с расширением .log).

В разделе «**Дальнейшие действия**» нажмите кнопку «**Обзор…**» и выберите логи отработки динамического анализа, чтобы загрузить логи отработки датчиков в проект.

Логи должны находиться в архиве с расширением .zip.

При необходимости имеется возможность задать кодировку zip-архива. После этого нажмите кнопку **«Начать загрузку».**

При этом статус проекта должен измениться на «Обработка результатов динамического анализа».

Необходимо дождаться окончания динамического анализа. При необходимости следует нажимите кнопку «Обновить».

При окончании анализа статус проекта должен измениться на «Динамический анализ завершен». После этого на странице проекта в разделах «Отчеты» и «Журналы» появятся дополнительные кнопки для просмотра отчетов и журналов по динамическому анализу.

Чтобы посмотреть отчсет по динамическому анализу в разделе «Отчеты» нажмите кнопку «Отчеты по динамическому анализу».

6.3 Примеры отчетов

На рисунке 28 показана страница с отчетами по динамическому анализу.

РИСУНОК 28//Отчеты по динамическому анализу

Список отчётов по динамическому анализу проекта "Operator_guide_trial"

- Отчёт по метрикам
- Отработавшие ФО (процедуры и функции)
- Отработавшие связи между ФО (процедурами и функциями)
- Отработавшие ФО (ветви)
- Отработавшие связи между ФО (процедурами, функциями и ветвями)

Чтобы посмотреть интересующий отчет, необходимо навести на него курсор и нажать левую кнопку мыши. Примеры отчетов представлены на рисунках 29 – 39.

На рисунке 29 представлен **отчет по метрикам динамического анализа.**

РИСУНОК 29//Отчет по метрикам динамическорго анализа

Operator_guide_trial: отчёт по метрикам динамического анализа

Количество подтвердившихся вызовов ФО/общее количество ФО	0/279
Процент покрытия по ФО	0%
Количество подтвердившихся связей ФО/общее количество связей ФО	0/42
Процент покрытия по связям ФО	0%
Количество подтвердившихся вызовов ветвей/общее количество ветвей	0/399
Процент покрытия по ветвям	0%
Количество подтвердившихся связей ФО с ветвями/общее количество связей ФО с вет	0/416
Процент покрытия по связям ФО с ветвями	0%

На рисунке 30 представлены **отработавшие ФО (процедуры и функции)**. В отчете имеются следующие поля: **№, QID, Название и Отработал**. Красным цветом выделены ФО, не отработавшие в ходе динамического анализа, зеленым — отработавшие.

РИСУНОК 30//Отработавшие ФО (процедуры и функции)

Git: отработавшие ФО (процедуры и функции)

№	QID	Название	Отработал
1	0:8	cmd_capabilities	-
2	0:10	terminate_batch	-
3	0:11	read_ref_note	-
4	0:21	parse_rev_note	-
5	0:33	note2mark_cb	-
6	0:46	regenerate_marks	-
7	0:51	check_or_regenerate_marks	-
8	0:64	cmd_import	-
9	0:85	cmd_list	-
10	0:87	do_command	-
11	0:101	main	-
12	1:1	main	-
13	2:3	get_next	-
14	2:5	set_next	-
15	2:8	compare_strings	-
16	2:13	main	-
17	3:1	git_user_agent	+
18	3:5	git_user_agent_sanitized	+
19	4:0	check_removed	-

На рисунке 31 представлены **отработавшие связи между ФО (процедурами и функциями)**. В отчете имеются следующие поля: **№, QID вызывающего объекта, Название вызывающего объекта, QID вызываемого объекта, Название вызываемого объекта и тип связи.**

РИСУНОК 31//Отработавшие связи между ФО (процедурами и функциями)

Operator_guide_trial: отработавшие связи между ФО (процедурами и функциями)

№	QID вызывающего объекта	Название вызывающего объекта	QID вызываемого объекта	Название вызываемого объекта	Тип связи
1	0:34	spawn_and_wait	0:4	wait_for_process	S
2	1:159	test_issue_92	1:149	send_ascii_command	S
3	1:162	test_issue_102	1:149	send_ascii_command	S
4	1:173	shutdown_memcached_server	1:149	send_ascii_command	S
5	1:507	test_issue_101	1:252	test_binary_noop	S
6	5:104	worker_libevent	5:46	register_thread_initialized	S
7	5:107	thread_libevent_process	5:46	register_thread_initialized	S
8	5:107	thread_libevent_process	5:86	cqi_free	S
9	5:225	thread_init	5:41	wait_for_thread_registration	S
10	7:28	slabs_preallocate	7:56	do_slabs_newslab	S
11	7:56	do_slabs_newslab	7:48	split_slab_page_into_freelist	S
12	7:62	do_slabs_alloc	7:56	do_slabs_newslab	S
13	7:213	do_slabs_reassign	7:202	slabs_reassign_pick_any	S
14	11:478	server_socket_unix	11:470	new_socket_unix	S
15	11:556	main	11:26	stats_init	S
16	11:556	main	11:30	settings_init	S
17	11:556	main	11:32	conn_init	S
18	11:556	main	11:498	clock_handler	S
19	11:556	main	11:506	usage	S
20	11:556	main	11:508	usage_license	S
21	11:556	main	11:510	save_pid	S
22	11:556	main	11:541	sigignore	S
23	11:556	main	11:545	enable_large_pages	S

На рисунке 32 представлены **отработавшие ФО (ветви)**. В отчете имеются поля, аналогичные представленным в отработавших ФО (процедурах и функциях).

РИСУНОК 32//Отработавшие ФО (ветви)

Operator_guide_trial: отработавшие ФО (ветви)

№	QID	Название	Отработал
1	0:11	for	-
2	0:14	if	-
3	0:22	if	-
4	0:25	switch	-
5	0:27	if	-
6	0:29	switch	-
7	0:31	switch	-
8	0:32	if	-
9	0:39	switch	-
10	0:41	switch	-
11	0:43	switch	-
12	1:26	if	-
13	1:40	for	-
14	1:48	for	-
15	1:51	for	-
16	1:56	if	-
17	1:85	if	-
18	1:90	if	-
19	1:92	while	-
20	1:95	if	-
21	1:98	while	-
22	1:100	if	-
23	1:115	if	-
24	1:117	if	-
25	1:119	if	-

На рисунке 33 представлены **отработавшие связи между ФО (процедурами, функциями и ветвями)**. В отчете имеются поля, аналогичные представленным в отчете по отработавшим связям между ФО (процедурами и функциями).

РИСУНОК 33//Отработавшие связи между ФО (процедурами, функциями и ветвями)

973	10:117	http_cleanup	10:119	while	S
974	10:117	http_cleanup	10:122	if	S
975	10:117	http_cleanup	10:123	if	S
976	10:119	while	10:121	if	SD
977	10:124	get_active_slot	10:128	while	S
978	10:124	get_active_slot	10:130	while	S
979	10:124	get_active_slot	10:131	if	SD
980	10:124	get_active_slot	10:135	if	SD
981	10:124	get_active_slot	10:136	if	S
982	10:124	get_active_slot	10:137	if	S
983	10:128	while	10:129	if	S
984	10:129	if	10:36	process_curl_messages	S
985	10:131	if	10:132	if	SD
986	10:131	if	10:133	if	S
987	10:131	if	70:18	xmalloc	SD
988	10:133	if	10:134	while	S
989	10:137	if	10:69	init_curl_http_auth	S
990	10:138	start_active_slot	10:142	if	S
991	10:147	add_fill_function	10:152	while	S
992	10:153	fill_active_slots	10:155	while	SD
993	10:153	fill_active_slots	10:160	while	SD
994	10:155	while	10:157	for	S
995	10:155	while	10:159	if	SD
996	10:157	for	10:158	if	S
997	10:160	while	10:161	if	S
998	10:162	step_active_slots	10:165	do	SD
999	10:162	step_active_slots	10:166	if	SD

Чтобы скачать все отчеты по динамическому анализу:

В разделе **«Отчеты»** нажмите кнопку **«Скачать все отчеты»**. Отчеты скачиваются в архиве с расширением .zip.

Чтобы посмотреть отчеты динамического анализа:

Откройте в браузере файл **dynindex.html.**

Чтобы посмотреть журнал динамического анализа:

В разделе **«Журналы»** нажмите кнопку **«Журнал динамического анализа»**. Журнал выглядит так, как показано на рисунке 34.

РИСУНОК 34//Пример журнала динамического анализа

```
Starting project execution at 16:09:34 13/02/2015
JVM memory settings:
Total memory: 62390272
Free memory: 49967104
Maximum memory: 922746880
Launching dynamic report with arguments .././data/projects/1/prj -l 1 at 16:09:34 13/02/2015
Project completed successfully at 16:09:35 13/02/2015
```

Чтобы провести новый динамический анализ:

На странице проекта в разделе **«Что дальше»** нажмите кнопку **«Новый динамический анализ»**.

7 Конфигурация парсеров

Загрузите файлы с настройками парсеров в виде zip-архива через веб-интерфейс.

Файл с настройками парсера имеет название “**parser-XXX**”,

где XXX - сокращенное название языка (например, cpp, java, php) и содержится в директории config.

Например, для парсера C++ в zip-архиве будет директория config, а в ней файл parser-cpp.

7.1 Конфигурация парсера C/C++

Укажите предустановленные макросы и пути к заголовочным файлам для проведения корректного анализа проектов на C/C++.

Файл конфигурации парсера хранится в XML-виде и позволяет задать:

- общие настройки парсера;
- аргументы парсера всех файлов и для каждого файла в отдельности.

Формат XML в этом случае имеет следующую структуру: корневой элемент – config, дочерние элементы – settings и arguments.

Arguments хранит аргументы парсера в виде линейного списка из argument.

Settings хранит параметры парсера в виде линейного списка из param. Каждая настройка хранится в теге param, значение атрибута name задает название параметра, value - значение. Пример изображен на рисунке 35.

РИСУНОК 35//Пример задания настроек

```
<settings>
        <param name>="threadCount"  value="0"  />
        < param name>="usePredefines"  value="0"  />>
</settings>
```

Перечень доступных параметров представлен в таблице 2.

Таблица 2. Доступные параметры.

Название	Описание	Возможные значения	Значение по умолчанию
`threadCount`	количество потоков	• `-1` программа сама вычисляет необходимое количество • `0` - не использовать потоки, • `n(n>0)` - количество потоков	-1
`useAutoTypeForReturn`	использовать тип «auto» для временной переменной выражения под return	• `0` - не использовать (вычисляется возвращаемый тип функции) • `1` - использовать	0
`usePredefines`	загружать предустановленные макросы ОС и компилятора	• `0` - не загружать • `1` - загружать	1

Через аргументы можно задать **include-path, define** и пр. Аргументы задаются с помощью тега argument так, как показано на рисунках 36 и 37. Если атрибуты **field** и **path** отсутствуют, то данная опция будет распространяться на все файлы. На рисунке 38 приведен пример конфигурационного файла.

РИСУНОК 36//Пример задания аргумента

```
<argument fileId="1" value="-I/include/path/"  >
```

РИСУНОК 37//Пример задания аргумента

```
<argument path="abc/1.cpp" value="-I/include/path/"  >
```

РИСУНОК 38//Пример содержимого конфигурационного файла

```
<?xml version="1.0"?>
<config>
        <arguments>
                <argument value="-DSOMEDEFINE"  />
                <argument fileId="1" value="-I/home/user/include/a"  />
                <argument fileId="2" value="-I/home/user/include/b"  />
        </arguments>

</config>
```

7.2 Конфигурация парсера PHP

Файл конфигурации парсера хранится в XML-виде и имеет следующую структуру: корневой элемент – config, дочерний элемент – settings.

Settings хранит параметры парсера в виде линейного списка из param. Каждая настройка хранится в теге param, значение атрибута name задает название параметра, value – значение. Список параметров представлен в таблице 3.

Таблица 3. Список параметров.

Параметр	Описание	Возможные значения	Значение по умолчанию
short_tags	Доступны «короткие» теги	0 или 1	1
asp_tags	Доступны теги в стиле ASP	0 или 1	0

РИСУНОК 39//Пример содержимого конфигурационного файла

```
<?xml version="1.0"?>
<config>
        <settings>
                <param name="short_tags" value="1"  />
                <param name="asp_tags" value="0"  />
        </settings>

</config>
```

7.3 Конфигурация парсера C#

Файл конфигурации парсера хранится в XML-виде и имеет следующую структуру: корневой элемент – config, дочерний элемент – settings.

Settings хранит параметры парсера в виде линейного списка из param. Каждая настройка хранится в теге param, значение атрибута name задает название параметра, value – значение. Список параметров представлен в таблице 4.

Таблица 4. Список параметров.

Параметр	Описание	Возможные значения	Значение по умолчанию
`version`	Версия стандарта C#	1.0, 2.0, 3.0, 4.0, 5.0	5.0
`astCountInMemory`	Количество хранимых AST в памяти. Можно увеличить это значение при большом количестве оперативной памяти	Целое число, отрицательное число означает хранить все AST в памяти	5000

РИСУНОК 40//Пример конфигурационного файла

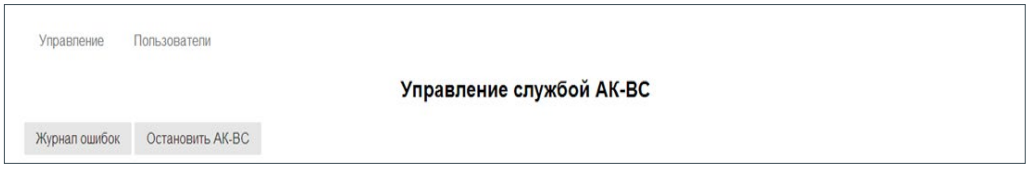
```
<config>
        <settings>
                <param name="version" value="4.0"  />

        </settings>

</config>
```

8 Администрирование «АК-ВС 2

Административная часть интерфейса доступна только пользователю **admin** по ссылке **127.0.0.1:11000/admin**

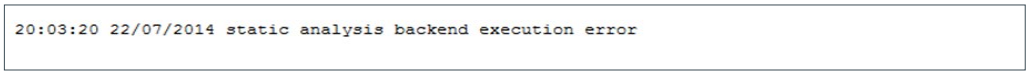
Административная часть веб-интерфейса выглядит, как показано на рисунке 41.

РИСУНОК 41//Административная страница веб-интерфейса



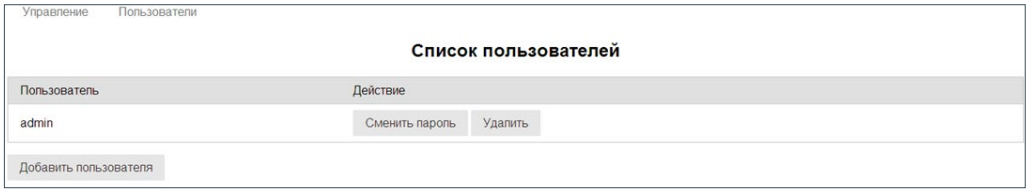
На вкладке «**Управление**» имеется возможность посмотреть журнал ошибок (см. рис. 42) и остановить работу «АК-ВС 2», нажав соответствующие кнопки.

РИСУНОК 42//Пример журнала ошибок



На вкладке «**Пользователи**» (см. рис. 43) отображается список всех пользователей «АК-ВС 2».

РИСУНОК 43//Список пользователей



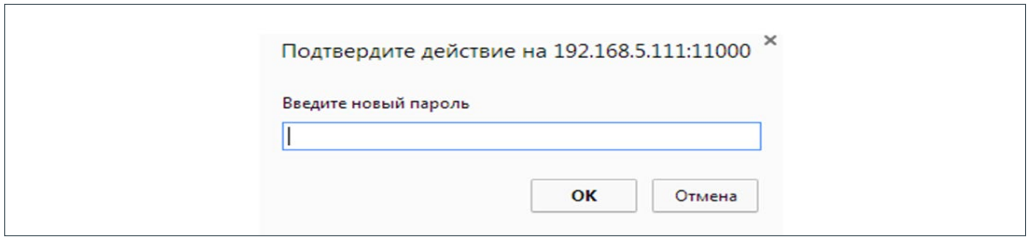
Чтобы удалить существующего пользователя:

Нажмите кнопку «**Удалить**» напротив удаляемого пользователя.

Чтобы сменить пароль существующего пользователя:

Нажмите кнопку «**Сменить пароль**» напротив выбранного пользователя. После этого появится окно ввода, изображенное на рисунке 44.

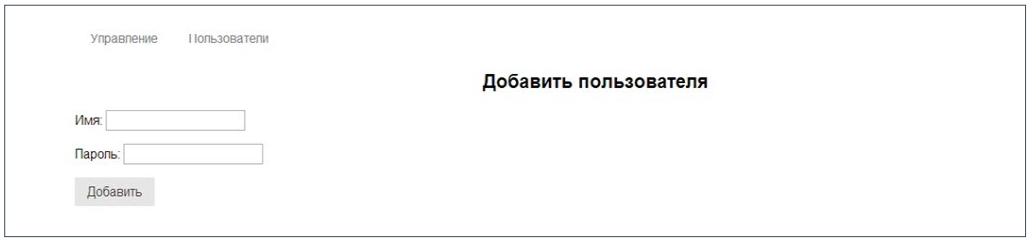
РИСУНОК 44//Окно ввода нового пароля



После нажатия кнопки «**OK**» появится окно, подтверждающее успешное обновление пароля.

Чтобы добавить нового пользователя нажмите кнопку «**Добавить пользователя**». После этого появится окно добавления нового пользователя, изображенное на рисунке 45

РИСУНОК 45//Окно добавления нового пользователя



В предложенных формах введите имя и пароль пользователя, после чего нажмите кнопку «**Добавить**», и новый пользователь добавится в список пользователей.

Чтобы завершить работу программмы нажмите кнопку «**Остановить АК-ВС 2**», в окне «**Управление службой АК-ВС**», доступной по ссылке http://127.0.0.1:11000/admin#service.

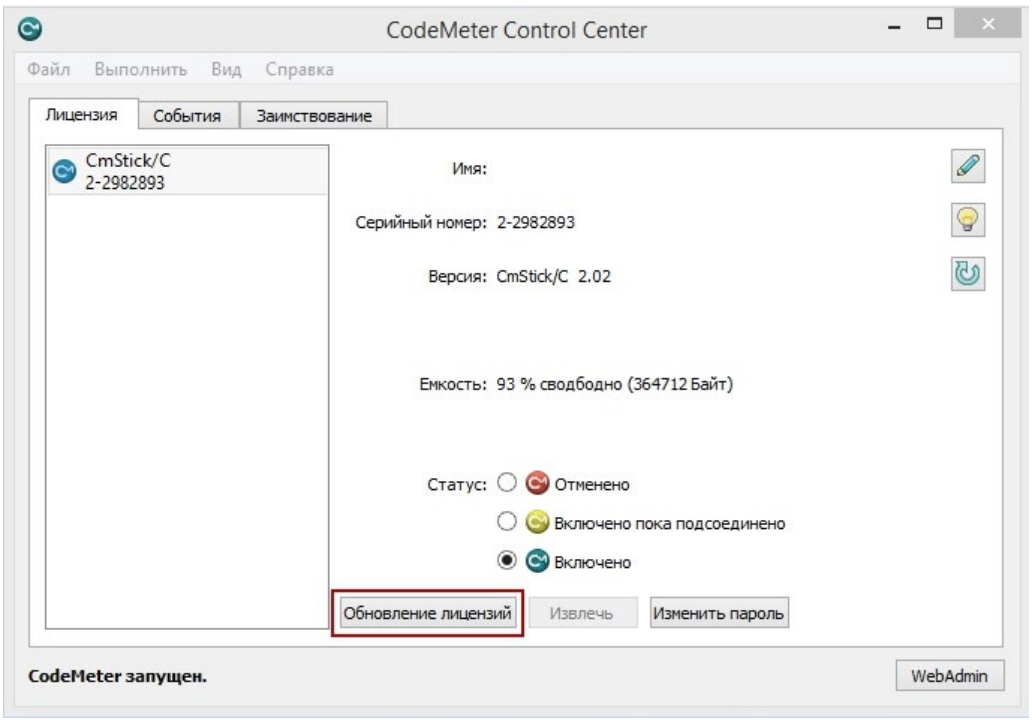
Приложение А

Обновление лицензии на операционные системы семейства Windows

Обновление лицензии происходит в два этапа. На первом этапе формируется запрос лицензии, на втором этапе импортируется полученный от разработчиков ПО файл обновления.

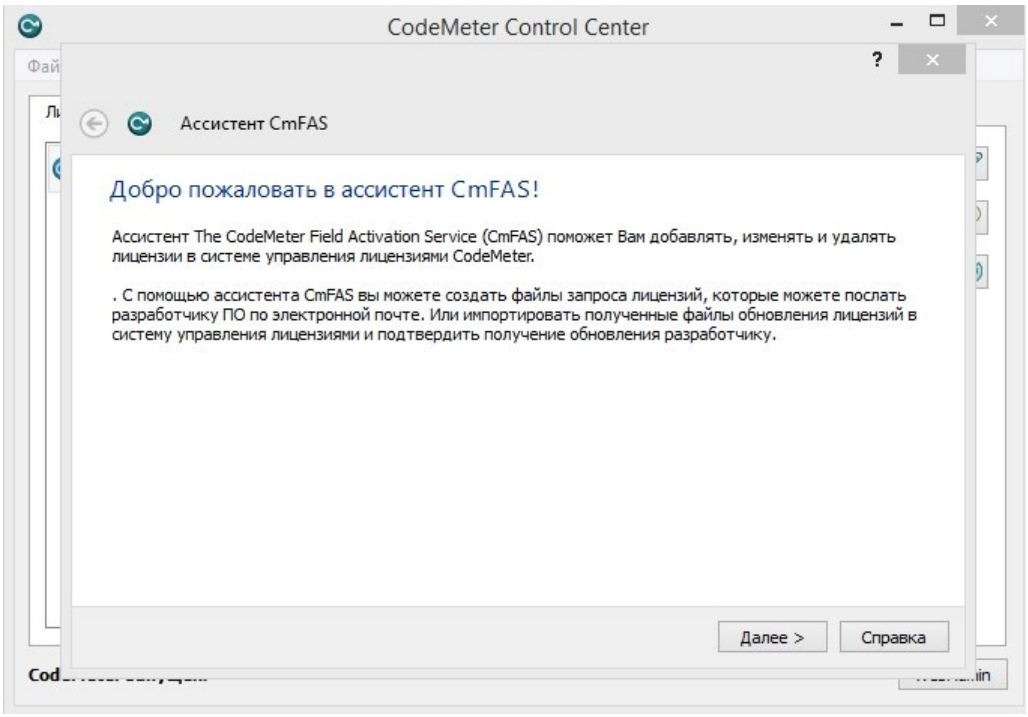
Запустите CodeMeter Control Center. Выберите лицензию, далее нажмите кнопку «**Обновление лицензии**»

РИСУНОК А.1//Окно CodeMeter Control Cente



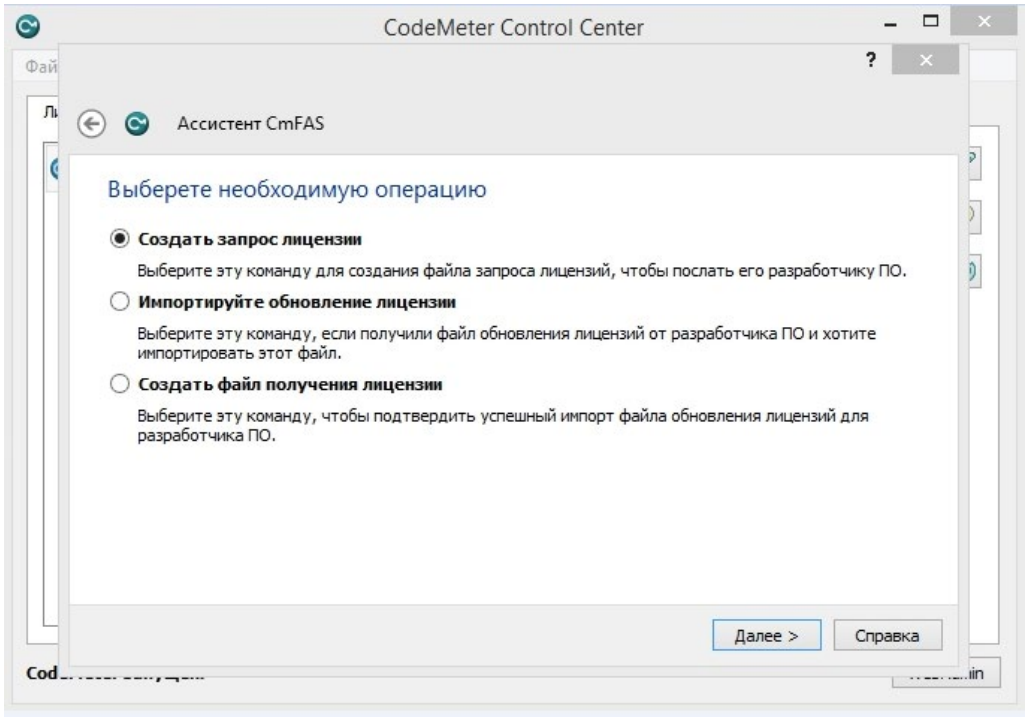
Появится окно с информацией об ассистенте CmFas.
Нажмите кнопку «**Далее**».

РИСУНОК А.2//Окно ассистента CmFas



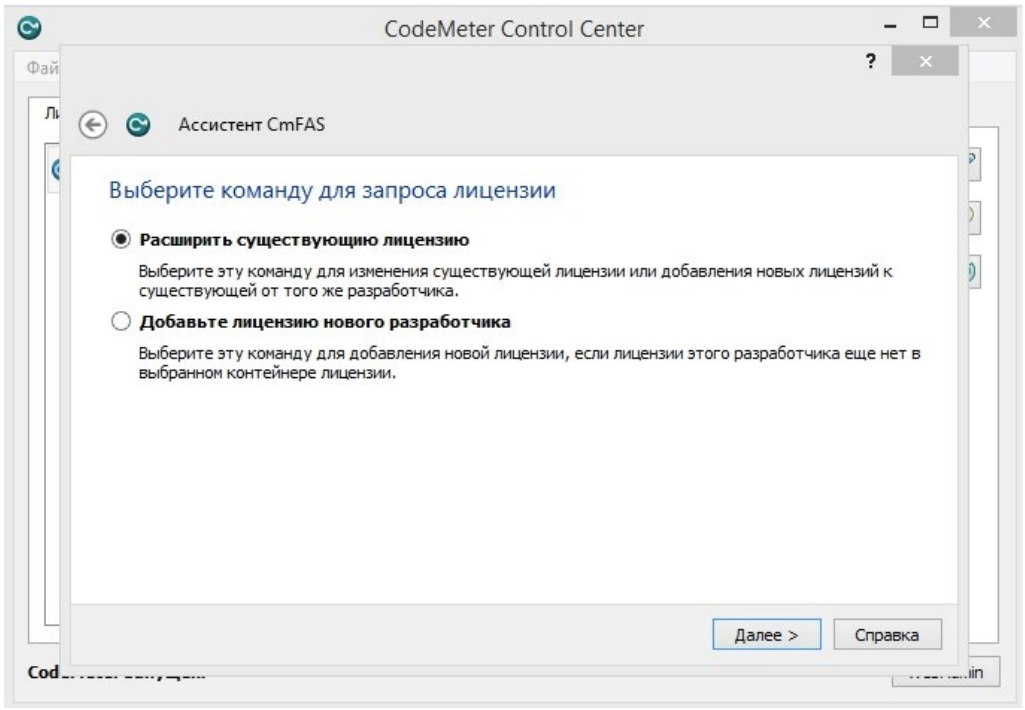
Выберите операцию «**Создать запрос лицензии**».
Нажмите «**Далее**».

РИСУНОК А.3//Окно выбора операции



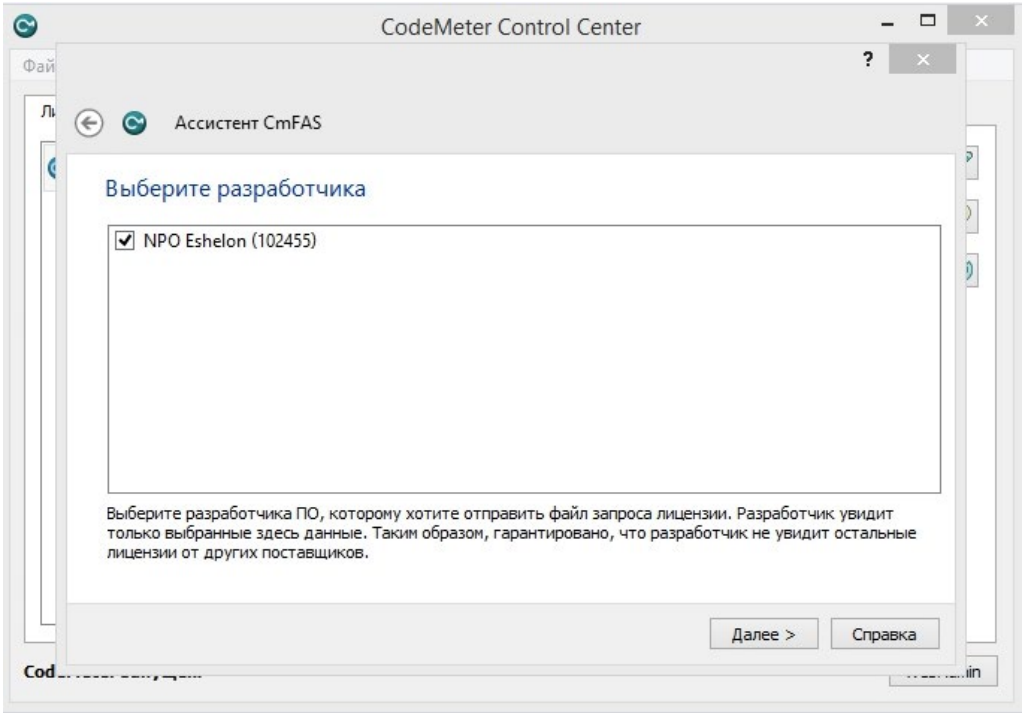
Выберите команду «**Расширить существующую лицензию**». Нажмите «**Далее**».

РИСУНОК А.4// Окно выбора команды для запроса лицензии



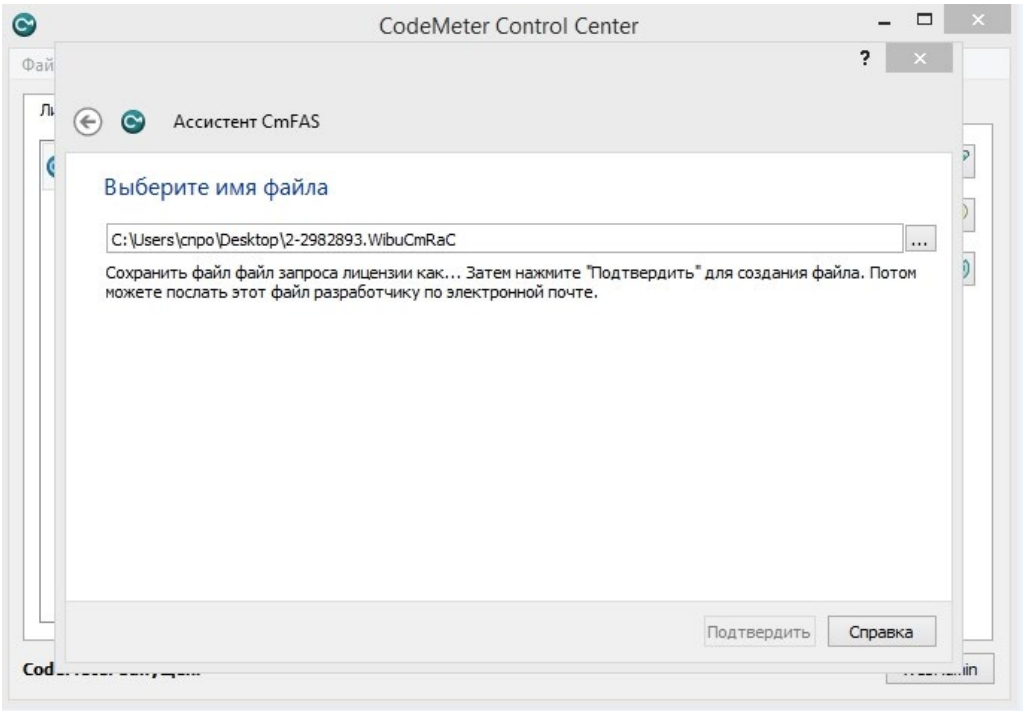
Выберите разработчика «NPO Eshelon». Нажмите «**Далее**».

РИСУНОК А.5//Окно выбора разработчика



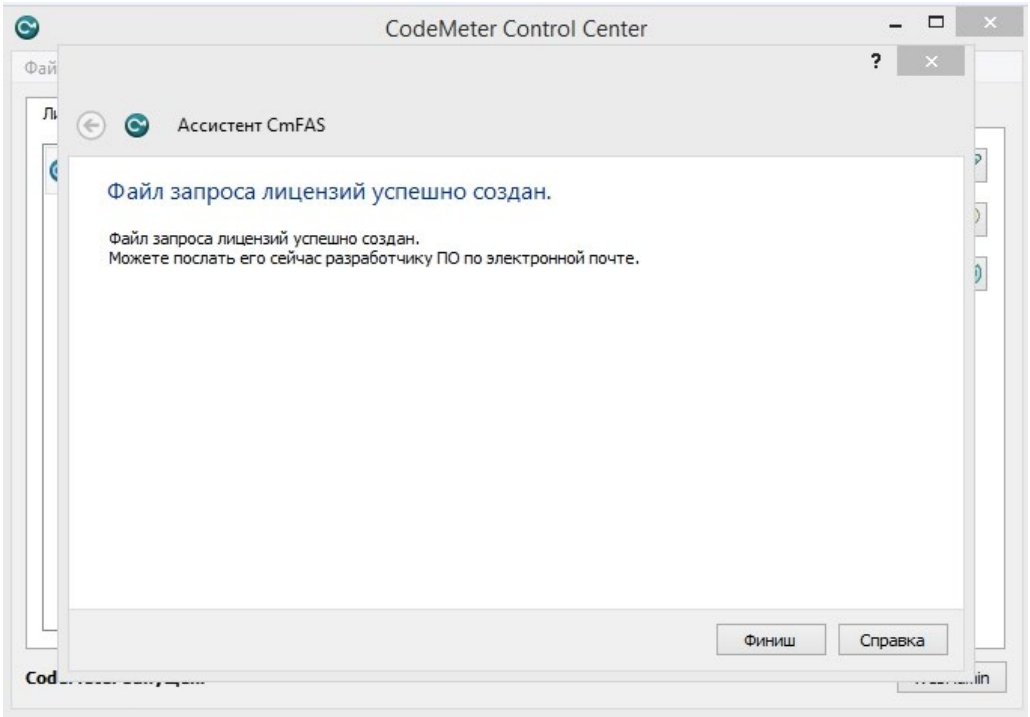
Укажите путь до файла, куда будет сохранен запрос.

РИСУНОК А.6//Окно выбора имени файла



Отправьте сформированный файл в ЗАО «НПО «Эшелон».

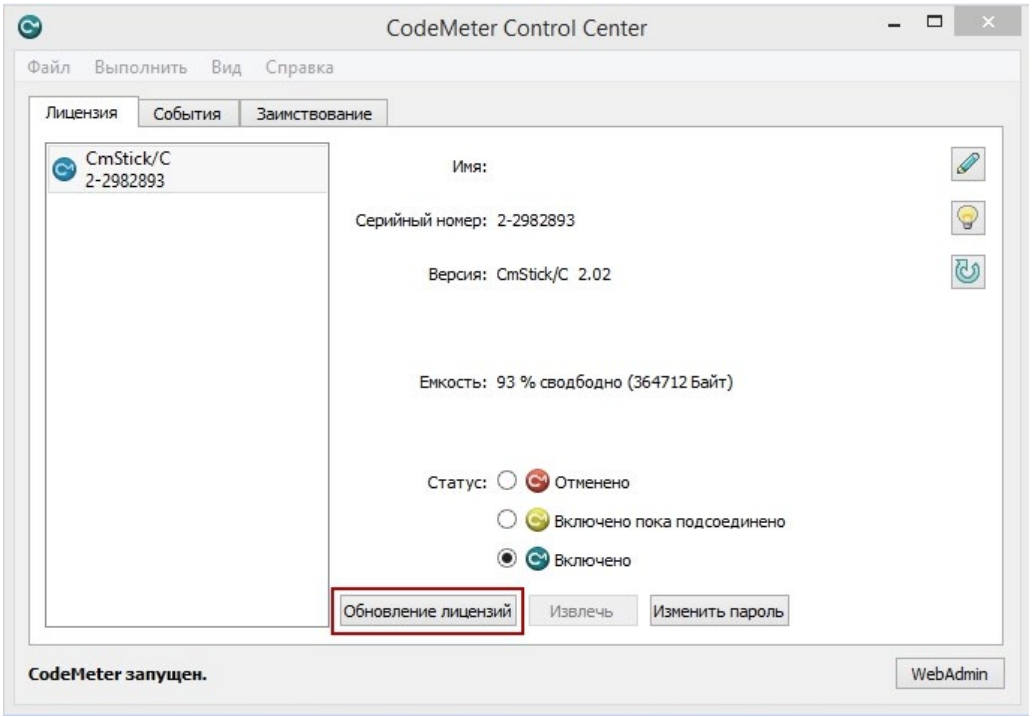
РИСУНОК А.7//Окно подтверждения успешного создания файла



После получения ответа от разработчика вновь запустите CodeMeter Control Center.

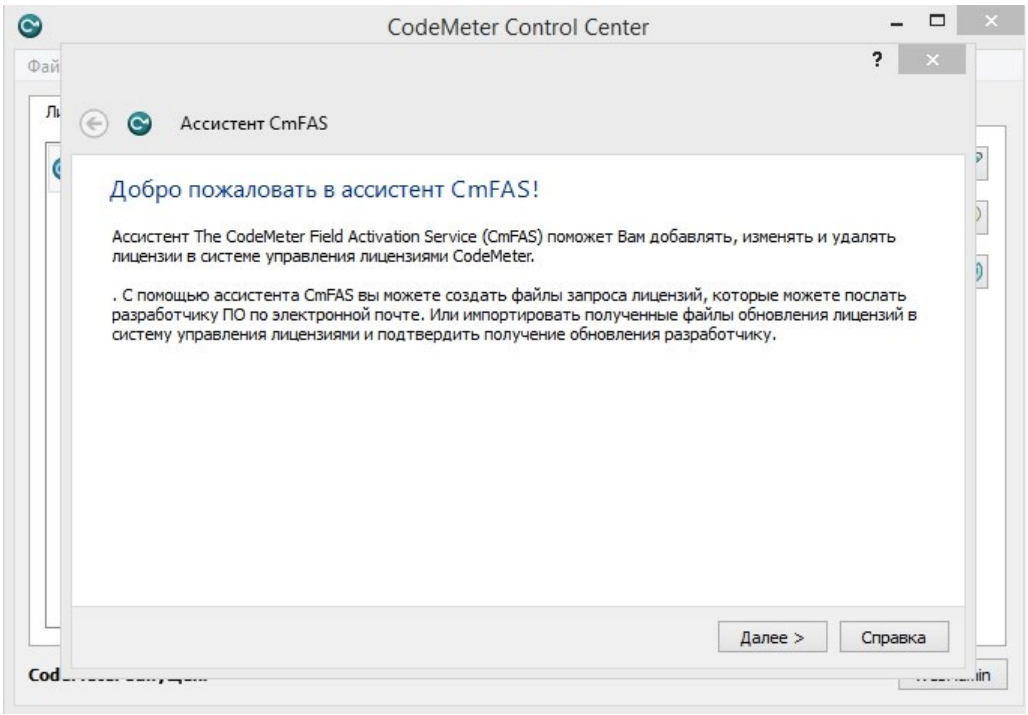
Выберите лицензию, далее нажмите кнопку «**Обновление лицензии**» .

РИСУНОК А.8//Окно CodeMeter Control Cente



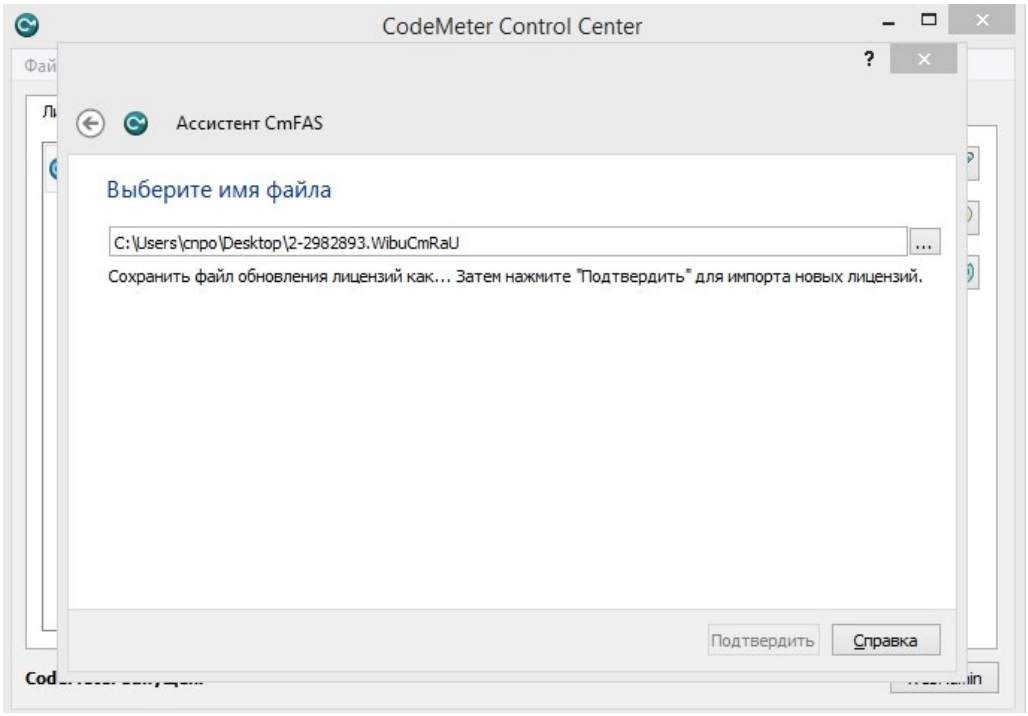
Нажмите кнопку «**Далее**».

РИСУНОК А.9//Окно ассистента CmFas



Выберите операцию «**Импортируйте обновление лицензии**».
Нажмите кнопку «**Далее**».

РИСУНОК А.10//Окно указания пути до полученного файла



После завершения всех действий лицензия будет обновлена.

Приложение Б

Обновление лицензии на операционные системы семейства Linux

Обновление лицензии происходит в два этапа. На первом этапе формируется запрос лицензии, на втором этапе импортируется полученный от разработчиков ПО файл обновления.

Выполните команду cmu --context 102455 --file 1.WibuCmRaC.

В результате выполнения команды будет сформирован файл запроса лицензии. Отправьте созданный файл в в ЗАО «НПО «Эшелон».

После ответа разработчика загрузите полученный файл с помощью команды cmu --import –file имя_файла.

Например, для файла с именем 1.WibuCmRaU команда будет выглядеть следующим образом: cmu --import --file 1.WibuCmRaU.

После завершения всех действий лицензия будет обновлена.